

Manual de utilización del cliente de Mentes con ATLAS

Versión 1.4

Arquitectura de Software



icm

Agencia de Informática y Comunicaciones de la Comunidad de Madrid



Hoja de Control

Título	Manual de Usuario de Utilización del cliente de Mentes		
Documento de Referencia	NORMATIVA ATLAS		
Responsable	ICM		
Versión	1.4	Fecha Versión	27/10/2015
¹Revisado/Validado por:		Fecha Revisión/Validación	
²Aprobado por:		Fecha Aprobación	

Registro de Cambios

Versión	Causa del Cambio	Responsable del Cambio	Fecha del Cambio
1.0	Versión inicial del documento	Arquitectura de Software	01/04/2013
1.1	Añadir el import de la configuración del fichero applicationContext-services.xml	Arquitectura de Software	03/02/2014
1.2	Actualizada su configuración para el framework ATLAS	Arquitectura de Software	05/08/2015
1.3	Actualizada la documentación	Arquitectura de Software	15/10/2015
1.4	Actualización de dependencias necesarias..	Arquitectura de Software	27/10/2015

Índice

1	INTRODUCCIÓN	4
1.1	AUDIENCIA OBJETIVO	4
1.2	CONOCIMIENTOS PREVIOS	4
2	DESCRIPCION DEL CLIENTE DE MENTES.	5
2.1	DEFINICIÓN Y DESCRIPCIÓN DE LA INTERFAZ DEL CLIENTE DE MENTES:	5
3	INSTALACION Y CONFIGURACION ATLAS.	14
3.1	INCLUSIÓN DE LA DEPENDENCIA:	14
3.2	CONFIGURACIÓN DEL ENDPOINT PARA EL CLIENTE.	15
3.3	INCLUSIÓN DEL CONTEXTO DEL CLIENTE EN APLICACIÓN ATLAS:	15
3.4	INCLUSIÓN DEL CLIENTE EN EL CONTEXTO DE SPRING:	16
3.5	INYECCIÓN DEL SERVICIO DE MENTES ATLAS.	17
3.6	INCLUSIÓN DEL CONTEXTO DEL CLIENTE EN TEST UNITARIOS JUNIT:	17
4	EJEMPLO DE UTILIZACION DEL CLIENTE	18
4.1	ENVIO SIMPLE:	19
4.2	ENVIO MULTIPLE (SIMPLE):	19
4.3	ENVIO MULTIPLE:	20
4.4	EJEMPLO DE USO DE LA CLASE DATOSENVIOSMS.	21
5	PREGUNTAS MÁS FRECUENTES	22
5.1	EN EL ENTORNO DE DESARROLLO NO ENVIA LOS MENSAJES SMS	22

Contenido

1 INTRODUCCIÓN

Para realizar envíos y recepción de mensajes de texto aplicaciones de la Comunidad de Madrid, se ha optado por una solución centralizada que a través de una plataforma de Telefónica (MENTES) . Esta plataforma realiza los envíos y registra las recepciones de mensajes de texto de forma independiente, de forma que todos los envíos se centralizan en un único servicio Web, pero las recepciones y el procesado de las mismas se deben implementar de forma específica por cada aplicación siguiendo las instrucciones del documento de Recepción de SMS's.

El acceso a esta plataforma esta restringido, solo se podrá interactuar con ella a través de un servicio Web. Este servicio Web solicita un identificador de aplicación para realizar envío de mensajes ya que no ninguna aplicación puede realizar envíos de mensajes sin solicitar previamente acceso a este servicio.

1.1 AUDIENCIA OBJETIVO

Este documento está orientado a todos los desarrolladores que desea realizar envíos de mensajes de texto en una aplicación desarrollada para la Comunidad de Madrid, en tecnología java, concretamente en ATLAS.

Con el presente documento se pretende ofrecer una guía completa de configuración y utilización del cliente del servicio Web de envío de mensajes de texto.

En los siguientes apartados se describen las diferentes peticiones que permite realizar el cliente a la plataforma (a través del servicio Web), siendo estas peticiones tanto de realización de envíos como de consulta del estado de los mismos.

Este documento se centra en el envío de mensajes y en la consulta de los estados de los envíos solicitados, la recepción y tratamiento de mensajes recibidos no se realiza a través del cliente, deben realizarse conforme al manual de Recepción de SMS's existente.

1.2 CONOCIMIENTOS PREVIOS

Para un completo entendimiento del documento, el lector deberá tener conocimientos previos sobre las siguientes tecnologías:

- Lenguaje Java
- Invocación de Servicios Web en Java
- Spring

2 DESCRIPCION DEL CLIENTE DE MENTES.

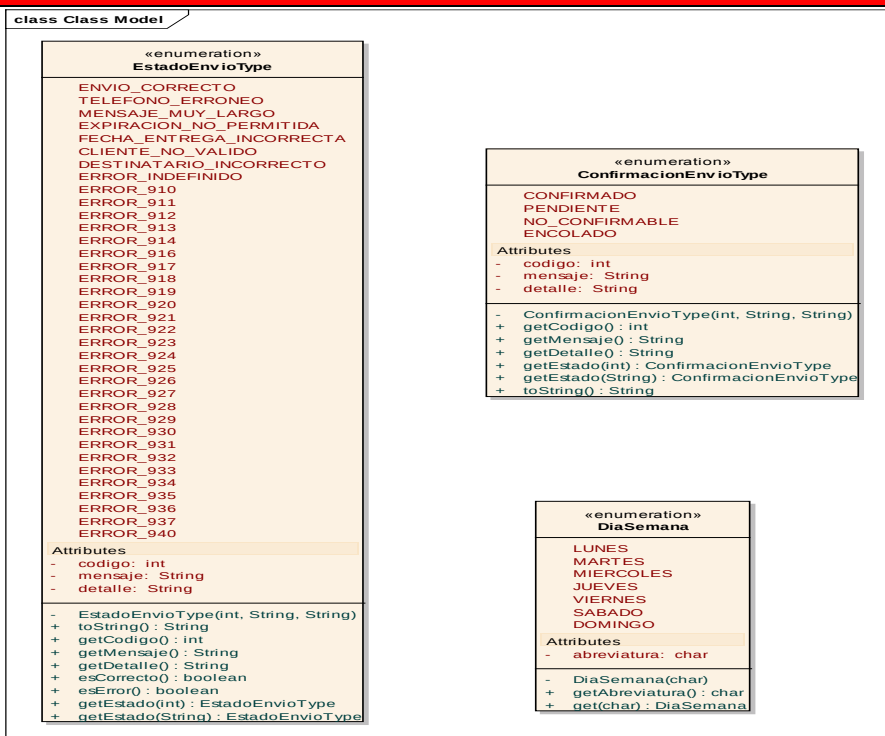
Como se ha comentado en la introducción, el envío de mensajes de texto por parte de aplicaciones de la Comunidad de Madrid debe realizarse a través de la plataforma MENTES, previa solicitud de autorización de cada una de las aplicaciones. Para normalizar la utilización de la plataforma y controlar su utilización, se ha creado un servicio Web con el que se pueden realizar las solicitudes de envío y las consultas necesarias para conocer el estado de los envíos solicitados. En tecnologías anteriores (FW2, Delphi) se permitía la creación de un cliente para invocar a este servicio Web, pero con ATLAS se ha creado un cliente único que deben utilizar todas las aplicaciones para realizar las operaciones con la plataforma. Este cliente proporciona los métodos necesarios para realizar todas las solicitudes de envío, así como las consultas de estado de los envíos solicitados, basándose en una serie de clases con las que se envía la información necesaria para realizar cada una de estas operaciones. A continuación se van a describir tanto los métodos disponibles en el cliente como las clases auxiliares con las que se envía la información.

2.1 Definición y descripción de la interfaz del cliente de MENTES:

El cliente del servicio Web de MENTES tiene los métodos necesarios para realizar envíos simples y complejos, es decir, permite enviar un mensaje simple a un teléfono simplemente indicando la aplicación, el mensaje y el teléfono de destino, enviar un mensaje a varios teléfonos indicando la aplicación, el mensaje y la lista de teléfonos, así como enviar múltiples mensajes a múltiples destinos con planificación o sin ella, con confirmación o sin ella. Ante la necesidad de manejar información de planificación, estado de envío, estado de confirmación, así como la necesidad de obtener un resumen de un envío complejo, se han utilizado una serie de clases definidas específicamente para ello, de esta forma se evita que cada uno gestione esta información como quiera y facilitando la comunicación entre el cliente y el servicio.

A continuación se describen todos estos tipos y las clases generados para manejar la información necesaria para realizar las solicitudes tanto de envío como de consulta de estado de envíos:

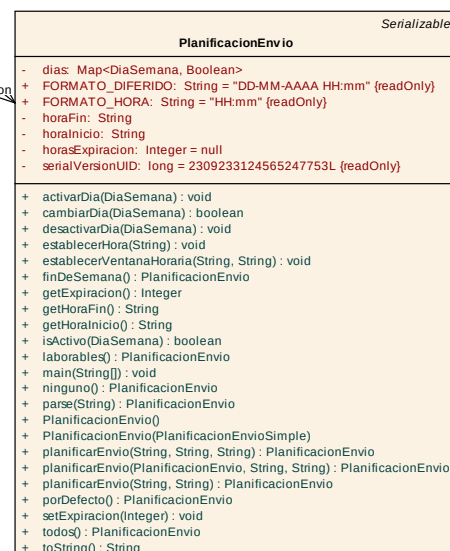
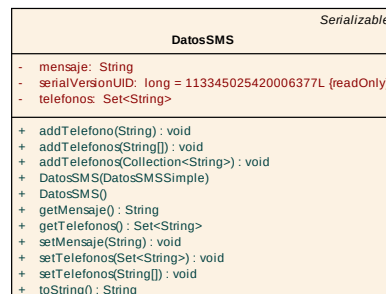
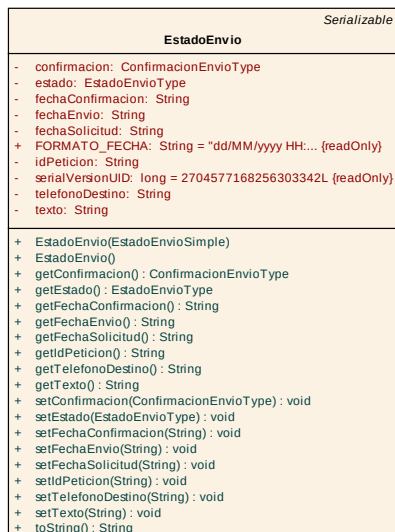
Tipos enumerados:



EstadoEnvioType	Tipo enumerado que define los posibles estados en los que se puede encontrar un envío. Este tipo además contiene la lista con los posibles errores que se pueden producir al realizar un envío.
ConfirmacionEnvioType	Tipo enumerado que define los posibles estados de confirmación de un envío.
DiaSemana	Tipo enumerado que define los días de la semana.

Clases:

class Class Model



DatosEnvioSMS	Descripción:		
	Clase con la información necesaria para realizar un envío complejo.		
	Atributos:		
	Nombre	Tipo	Descripción
	Aplicación:	String	Identificador de la aplicación que realiza el envío.
	CodigoCliente:	String	Código de cliente del envío.
	Confirmación:	boolean	Indicador de si el envío requiere confirmación.
	DatosSMS:	Collection <DatosSMS>	Lista de mensajes a enviar con formato de DatosSMS.
	Planificación:	PlanificacionEnvio	Información de la planificación.
	TipoServicio:	String	Tipo de servicio del envío.
	Métodos(no se incluyen métodos getter y setter ni constructores):		
	addMensaje (mensaje String, telefono String):		Método que incluye un mensaje a un teléfono.
	addMensaje (mensaje String, telefonos Collection<String>):		Método con el que se incluye un mensaje a varios teléfonos.
	addMensaje (datosSMS DatosSMS):		Método con el que se incluye un envío completo.
	planificarDiaSemana (planificacion PlanificacionEnvio)		Método con el que se establece la planificación del envío a través de un objeto PlanificacionEnvio.
DatosSMS:	Descripción:		
	Clase que almacena un mensaje y la lista de teléfonos a la que hay que enviar el mensaje.		
	Atributos:		
	Nombre	Tipo	Descripción
	Mensaje:	String	Texto a enviar.
	Telefonos:	String	Lista de teléfonos de destino.
	Métodos(no se incluyen métodos getter y setter ni constructores):		
	addTelefono(telefono String)		Método con el que se incluye un telefono en la lista de destinos del mensaje.
	addTelefono(telefono String[])		Método con el que se incluye un array de telefonos en la lista de destinos del mensaje.
	addTelefono(telefono Collection<String>)		Método con el que se incluye una colección de telefonos en la lista de destinos del mensaje.

Descripción:		
Clase con la que se define una planificación de un envío.		
Atributos:		
Nombre	Tipo	Descripción
Dias:	Map <DiasSemana, boolean>	Lista con los días de la semana y un indicador de envío para cada día.
Horainicio:	String	Hora de inicio de la planificación.
HoraFin:	String	Hora de fin de la planificación.
HorasExpiracion:	Integer	Número de horas de expiración del mensaje.
Métodos(no se incluyen métodos getter y setter ni constructores):		
activarDia(dia DiasSemana)		Método con el que se activa el día de la semana recibido como parámetro.
desactivarDia(dia DiasSemana)		Método con el que se desactiva el día de la semana recibido como parámetro.
cambiarDia(dia DiasSemana)		Función con la que se cambia el estado del día recibido como parámetro, devolviendo el estado en el que queda el día.
establecerHora(hora String)		Método con el que se establece la hora del envío, estableciendo la hora de inicio y la de fin con el valor del parametro recibido.
establecerVentanaHoraria(horainicio String, horaFin String)		Método con el que se establece una ventana horaria.
finDeSemana()		Función estática que devuelve un objeto PlanificacionEnvio con los días sábado y domingo activos y el resto inactivos.
Laborables()		Función estática que devuelve un objeto PlanificacionEnvio con los días de Lunes a Viernes activos y el sábado y el domingo inactivos.
porDefecto()		Función estática que devuelve un objeto PlanificacionEnvio con la planificación por defecto, los días de Lunes a Viernes activos y el sábado y el domingo inactivos.
todos()		Función estática que devuelve un objeto PlanificacionEnvio con los todos los días de activos.

EstadoEnvio

Descripción:		
Clase con la que manejar la información del estado de un envío.		
Atributos:		
Nombre	Tipo	Descripción
IdPetición	String	Identificador de la petición.
Confirmacion:	String	Indicador de si el envío requiere confirmación.
Estado:	EstadoEnvioType	Estado en el que se encuentra el envío.
FechaConfirmacion:	String	Fecha en la que se confirma el envío.
FechaEnvio:	String	Fecha en la que se realiza el envío.
FechaSolicitud:	String	Fecha en la que se solicita el envío.
TelefonoDestino:	String	Teléfono al que se envía el mensaje.
Texto:	String	Mensaje enviado.
Métodos(no se incluyen métodos getter y setter ni constructores):		
-	-	-

ResumenEnvio

Descripción		
Clase con la que se obtiene el resumen de un envío.		
Atributos		
Nombre	Tipo	Descripción
Aplicacion:	String	Aplicación que realiza el envío.
CdCliente:	String	Código de cliente del envío.
envioCompleto:	Boolean	Indicador de envío completo.
fechaEnvio:	String	Fecha en la que se realiza el envío.
IdPetición:	Integer	Identificador de la petición.
TipoServicio:	String	Tipo de servicio solicitado en el envío.
NumMensajesEnviados:	Integer	Número de mensajes enviados.
NumMensajesConfirmados:	Integer	Número de mensajes confirmados.
NumMensajesSolicitados:	Integer	Número de mensajes solicitados.
NumMensajesPendientes:	Integer	Número de mensajes pendientes.
Métodos:		
-	-	-

Una vez definidos estos tipos y estas clases, se comprende mucho mejor cada uno de los métodos del cliente.

Ciente:

class Class Model

«interface»

CienteSMS

```
+ enviar(DatosEnvioSMS) : String
+ envioMultiple(String, String, Collection<String>) : String
+ envioSimple(String, String, String) : String
+ esEnvioCorrecto(Long, String) : boolean
+ findByAplicacionFecha(String, Date, String, String) : Collection<EstadoEnvio>
+ findByIdPeticion(Long, String) : Collection<EstadoEnvio>
+ findErroneos(Long, String) : Collection<EstadoEnvio>
+ findNoConfirmados(Long, String) : Collection<EstadoEnvio>
+ resumenEnvioAplicacionFecha(String, Date, String, String) : Collection<ResumenEnvio>
+ resumenEnvioIdPeticion(Long, String) : ResumenEnvio
```

Descripción		
Clase con la información necesaria para realizar un envío complejo, con o sin planificación.		
Atributos		
Nombre	Tipo	Descripción
Aplicacion:	String	Aplicación desde la que se realiza el envío.
CodigoCliente:	String	Código de cliente del envío.
Confirmacion:	Boolean	Indicador de si el envío requiere confirmación.
DatosSMS:	Collection<DatosSMS>	Información del envío.
Planificacion:	PlanificacionEnvio	Información de la planificación del envío.
TipoServicio:	String	Tipo de servicio del envío.
Métodos:		
enviar(datos DatosEnvioSMS):String		Función que realiza un envío cuyo contenido se define a través de un objeto de tipo DatosEnvioSMS devolviendo el identificador asignado a la petición.
envioMultiple(String aplicacion, String texto, Collection<String> telefonos):String		Función que realiza el envío del mensaje recibido como parámetro para la aplicación recibida como parámetro a la lista de teléfonos recibida como parámetro, devolviendo el identificador asignado a la petición.
envioSimple(String aplicacion, String texto, String telefono):String		Función con la que se realiza un envío desde una aplicación, con el texto y al teléfono todos recibidos como parámetros, devolviendo el identificador asignado a la petición.
esEnvioCorrecto(Long idPetition, String aplicacion):boolean		Función con la que se comprueba el estado de un envío a partir de su identificador y el de la aplicación desde la que se realizó el envío, devolviendo si el envío se ha realizado correctamente o no.
findByAplicacionFecha (String aplicacion, Date fechaEnvio, String tipoServicio, String codigoCentro) : Collection<EstadoEnvio>		Función con la que se obtienen todos los envíos realizados desde una aplicación en una fecha concreta. Este método devuelve una colección de EstadoEnvio.
findByIdPetition(Long idPetition, String aplicacion):		Función con la que se obtiene el estado del envío identificado por el id recibido como parámetro. Este método devuelve un estadoEnvio.
findErroneos(Long idPetition, String aplicacion):		Función con la que se obtiene la lista de mensajes erróneos de la petición identificada por el id recibido como parámetro. Este método devuelve una colección de EstadoEnvio.
findNoConfirmados(Long idPetition, String aplicacion):		Función con la que se obtiene la lista de mensajes no confirmados de la petición identificada por el id recibido como parámetro. Este método devuelve una colección de EstadoEnvio.
resumenEnvioAplicacionFecha(String aplicacion, Date fecha, String tipoServicio, String codigoCentro):		Función con la que se obtienen los resúmenes de los envíos realizados por la aplicación recibida como parámetro, en la fecha recibida como parámetro del tipo de servicio y código de centro también recibidos como parámetros. Esta función devuelve una colección de ResumenEnvio.
resumenEnvioIdPetition (Long idPetition, String aplicacion):		Función con la que se obtiene el resumen de la petición identificada por el id que recibe como parámetro.

Interfaz y clases del cliente de MENTES.

class Class Model



3 INSTALACION Y CONFIGURACION ATLAS.

A lo largo de este apartado se especifica como configurar el cliente del servicio Web de Mentes para su utilización en cualquier tipo de módulo Atlas.

Independientemente del tipo de módulo en el que se quiere utilizar el cliente, es necesario incluir la dependencia con este, configurar su contexto específico, definir el endpoint (URL de acceso al servicio) e inyectar el bean con el que vamos a invocarlo.

3.1 Inclusión de la dependencia:

Para incluir la dependencia, simplemente debemos modificar el fichero pom.xml de la aplicación incluyendo en la sección <dependencies> las siguientes líneas. Esta configuración es la misma para los diferentes tipos de módulos de ATLAS.

Pom.xml

```
...
<dependencies>
...
    <dependency>
        <groupId>atlasfrm</groupId>
        <artifactId>atlasfrm-sms-mentesws-lib</artifactId>
        <version>${atlasfrm-sms-mentesws-lib.version}</version>
    </dependency>
</dependencies>
...
```

3.2 Configuración del endpoint para el cliente.

El cliente debe “apuntar” a diferentes destinos en función del entorno en el que se va a ejecutar la aplicación, por ello debe incluirse una línea en todos los ficheros `enviroment.properties` del módulo en el que se desea utilizar el envío de mensajes. Las líneas a incluir en estos ficheros son:

Enviroment.properties
...
Configuración del servicio web de MENTES
SMSMenTesService.endpoint= http://desarrollo.madrid.org/mentes_ws/services/ServicioSMSMenTes
...

NOTA: La URL incluida en cada fichero varía. Los valores para los ficheros de todos los entornos excepto el de desarrollo debe definirlo paso a producción, en el de desarrollo se deben copiar las líneas anteriores. Esta configuración aplica a todos los tipos de módulos ATLAS.

3.3 Inclusión del contexto del cliente en aplicación ATLAS:

El cliente puede utilizarse en cualquier tipo de módulo ATLAS, por lo que a continuación se define como debe incluirse el contexto del cliente para cada uno de los tipos de módulo existentes de ATLAS.

3.4 Inclusión del cliente en el contexto de Spring:

Para incluir el contexto del cliente es necesario incluir en el fichero web.xml, en el context-param contextConfigLocation, incluir la siguiente línea marcada:

Web.xml
<pre>... <context-param> <description>Este parámetro indica la localización exacta de los ficheros de configuración de SPRING </description> <param-name>contextConfigLocation</param-name> <param-value> ... classpath:/conf/applicationContext-atlasfrm-sms-mentesws-lib.xml; </param-value> </context-param> ...</pre>

NOTA

Esta configuración solo aplica a los tipos de módulos que tienen fichero web.xml (Web y WebService).

3.5 Inyección del servicio de Mentes Atlas.

Para utilizar el cliente, debemos incluir el servicio **AtlasSMSService** en la el servicio propio de la aplicación, que quiera utilizar su funcionalidad.

En el ejemplo de abajo ejemploServicio es el servicio de la aplicación que quiere usar la funcionalidad de mandar sms

applicationContext-services.xml

```
...
<import
  resource="classpath:/conf/applicationContext-atlasfrm-sms-mentesws-lib.xml" />

<bean id="ejemploService" class="xxxx.services.ejemploServiceImpl"
  ...
  p:atlasSMSService-ref="AtlasSMSService" />
...
```

3.6 Inclusión del contexto del cliente en test unitarios junit:

En todos los test de clases en las que se utilice el cliente del servicio Web de Mentes debe incluirse en la anotación `@ContextConfiguration` la siguiente localización:

ejemploServiceTest.java

```
...
@ContextConfiguration(locations = {
  ...
  "classpath:/conf/applicationContext-atlasfrm-sms-mentesws-lib.xml "
})
...
```

NOTA

Esta configuración aplica a todos los tipos de módulos de ATLAS.

4 EJEMPLO DE UTILIZACION DEL CLIENTE

Para utilizar el cliente, debemos incluir el servicio AtlasSMSService en la el servicio propio de la aplicación, que quiera utilizar su funcionalidad y inyectar este servicio en el fichero applicationContext-services.xml

ejemploServiceImpl.java

```
...  
  
/**  
 * Servicio de Mentes de Atlas (se inyecta)  
 */  
  
private AtlasSMSService atlasSMSService;  
  
/**  
 * Método setter del servicio de mentes  
 * @param service  
 */  
  
public void setAtlasSMSService (AtlasSMSService atlasSMSService) {  
    this. atlasSMSService = atlasSMSService;  
}  
  
...
```

4.1 Envío simple:

ejemploServiceImpl.java

```
...  
  
    /** {@inheritDoc} */  
    @Override  
    public String envioEjemploSMS(String aplicacion, String mensaje,  
                                   String telefono) throws ServiceException {  
  
        try {  
            return atlasSMSService.envioSimple(aplicacion, mensaje, telefono);  
        } catch (Exception e) {  
            throw new ServiceException(e);  
        }  
    }  
  
...
```

4.2 Envío multiple (simple):

Uso del servicio de SMSMenTes

```
...  
  
    /** {@inheritDoc} */  
    @Override  
    public String envioEjemploMultipleSMS(String aplicacion, String mensaje,  
                                           List<String> telefonos) throws ServiceException {  
  
        try {  
            return atlasSMSService.envioMultiple(aplicacion, mensaje, telefonos);  
        } catch (Exception e) {  
            throw new ServiceException(e);  
        }  
    }  
  
...
```

4.3 Envío multiple:

Uso del servicio de SMSMenTes

```
...  
  
    /** {@inheritDoc} */  
    @Override  
    public String envioSmsMultipleEjemplo(DatosEnvioSMS datos)  
        throws ServiceException {  
  
        try {  
            return atlasSMSService.enviar(datos);  
        } catch (Exception e) {  
            throw new ServiceException(e);  
        }  
    }  
...  

```

4.4 Ejemplo de uso de la clase DatosEnvioSMS

DatosEnvioSMS

```
...  
  
/**  
 * Método que realiza el envío de un mensaje con formato DatosEnvioSMS.  
 * @throws ServiceException si ocurre algun error  
 */  
public void enviarMensajeComplejo() throws ServiceException {  
    try {  
        DatosEnvioSMS datos = new DatosEnvioSMS();  
        DatosSMS sms = new DatosSMS();  
        PlanificacionEnvio plan = PlanificacionEnvio.laborables();  
        Collection<DatosSMS> lista = new ArrayList<DatosSMS>();  
        sms.setMensaje("Mensaje de prueba.");  
        sms.addTelefono("646212223");  
        lista.add(sms);  
        datos.setDatosSMS(lista);  
        datos.setAplicacion("APP1");  
        datos.setCodigoCliente("CC");  
        datos.setConfirmacion(false);  
        datos.setPlanificacion(plan);  
        datos.setTipoServicio("TS");  
        smsFacade.envioSmsMultipleEjemplo(datos);  
  
    } catch (ServiceException e) {  
        this.enviadoSN = false;  
        this.enviado = AtlasFacesUtils.getStringFromBundle("sms.envioerror")  
            + " --> " + e.getMessage();  
        AtlasFacesUtils.storeOnRequestError(FacesContext.getCurrentInstance(),  
            "error.SMS", e);  
    }  
  
}  
  
...
```

5 PREGUNTAS MÁS FRECUENTES

5.1 En el entorno de desarrollo no envía los mensajes SMS

En el entorno de desarrollo la plataforma Mentes no realiza el envío real de los SMS sino que lo simula y registra la acción en sus tablas de base de datos almacenando la información de los datos que a enviar. Estos registros pueden consultarse en la aplicación de consulta de envío de SMS, entrando con credenciales ADMON/ADMON:

http://desarrollo.madrid.org/ment_consulta_web



desarrollo.madrid.org/ment_consulta_web/html/web/ConsultaCiaEnvio.icm?ESTADO_MENU=1

Agencia de Informática y Comunicaciones de la Comunidad de Madrid | Área de Integración y Arquitectura de aplicaciones
Consulta de Envío de SMS

Inicio > Home

Cerrar

> Consulta CIA_ENVIO

Consulta CIA_RESULTADO

(c) ICM-2008

Introduzca el código de la aplicación:

Introduzca parámetros clausula where:

Consultar

Consulta Tabla MENT_CIA_ENVIO	
ID	NUMBER(11)
IDPETICION	NUMBER(11)
CD_APLICACION	VARCHAR2(20)
T_DESTINO	VARCHAR2(100)
T_ORIGEN	VARCHAR2(20)
TEXTO	VARCHAR2(480)
FECHAENVIO	VARCHAR2(12)
EXPIRACION	VARCHAR2(3)
TIPO	VARCHAR2(12)
CABECERA	VARCHAR2(10)
ENVIADO	NUMBER(11)
CONFIRMACION	NUMBER
ENTREGADIFERIDA	VARCHAR2(12)
CLASE	NUMBER(1)
LUNES	NUMBER(1)
MARTES	NUMBER(1)
MIERCOLES	NUMBER(1)
JUEVES	NUMBER(1)
VIERNES	NUMBER(1)
SABADO	NUMBER(1)
DOMINGO	NUMBER(1)
HORAINICIO	VARCHAR2(5)
HORAFIN	VARCHAR2(5)
IDMO	VARCHAR2(50)
REINTENTADO	NUMBER(11)
INSERTADO	DATE
CD_CLIENTE	VARCHAR2(20)
CD_TIPOSERVICIO	VARCHAR2(20)

WS- MALA METEOROLÓGICA

MADRID CO-MUNDO DIGITAL

Copyright © Comunidad de Madrid.

Aviso Legal | Privacidad | Contacto | Accesibilidad

desarrollo.madrid.org/ment_consulta_web/html/web/MuestraListadoCiaEnvio.icm


 Agencia de Informática y Comunicaciones de la Comunidad de Madrid | Área de Integración y Arquitectura de aplicaciones
 Consulta de Envío de SMS

Inicio > Home | Cerrar

Consulta CIA_ENVIO

Consulta CIA_RESULTADO

(c) ICM-2008

Lista de Envios

ID	IDPETICION	T_DESTINO	T_ORIGEN	TEXTO	FECHAENVIO	EXPIRACION	TIPO	CABECERA	ENVIADO	CONFIRMACION	ENTR
970379	96603	626755663	217035	Usuario con DNI 03188052E, su contraseña ha sido restablecida. La nueva contraseña para acceder	01-09-2015	0			0	1	

Numero total de registros: 23

1 | 2 | 3

>



 MADRID COMUNITAT DIGITAL

Copyright © Comunidad de Madrid.

Aviso Legal | Privacidad | Contacto | Accesibilidad